AUSTRALIAN UNIX USERS GROUP NEWSLETTER

Editors Rave

Well, better late than never, this issue is a bit smaller than the last few. I waited for as long as I could before producing it in the (vain) hope that I might get some summaries from people who spoke at the last user meeting, and some stuff to publish about the world UNIX conference in Melbourne in october. I made numerous phone calls to various people who spoke in Sydney on July 2, but nothing has turned up.

Well, you can wait only so long. Incidentally Geoff Cole, from the Sydney University Computer Centre, asks that the people who attended lunch at the last meeting and have not payed do so immediately.

There are two enclosures with this issue (summarised below) that I ask (plead!!) with you to take action on as soon as possible.

The Software Catalogue

Work has commenced on production of the catalogue, and already a lot of stuff has been sifted during a first 'find out whats there' phase. What is now needed is input from all you people out there as to what information you would like from the catalogue. What sort of queries do you have about software for UNIX. What are your interests, how should they be grouped, what do you want to know about an item which may fulfill your demands?

The enclosed form asks for information to define the structure of the database. Please make an effort to complete and return it so that we can get a better idea of what is needed.

Subscriptions For Volume III Of AUUGN

I have, just yesterday, received payment from the last outstanding subscriber to AUUGN, only nine months late. It is obvious that people need a lot of warning when it comes to paying money. So here it is.

BE WARNED !!! SUBSCRIPTIONS ARE NOW DUE !!!

Enclosed with this issue is the invoice for volume III of AUUGN. Please start your respective administrative balls rolling. I want money, not order forms and other such paper warfare.

To those of you who have already sent next years money - I thank you.

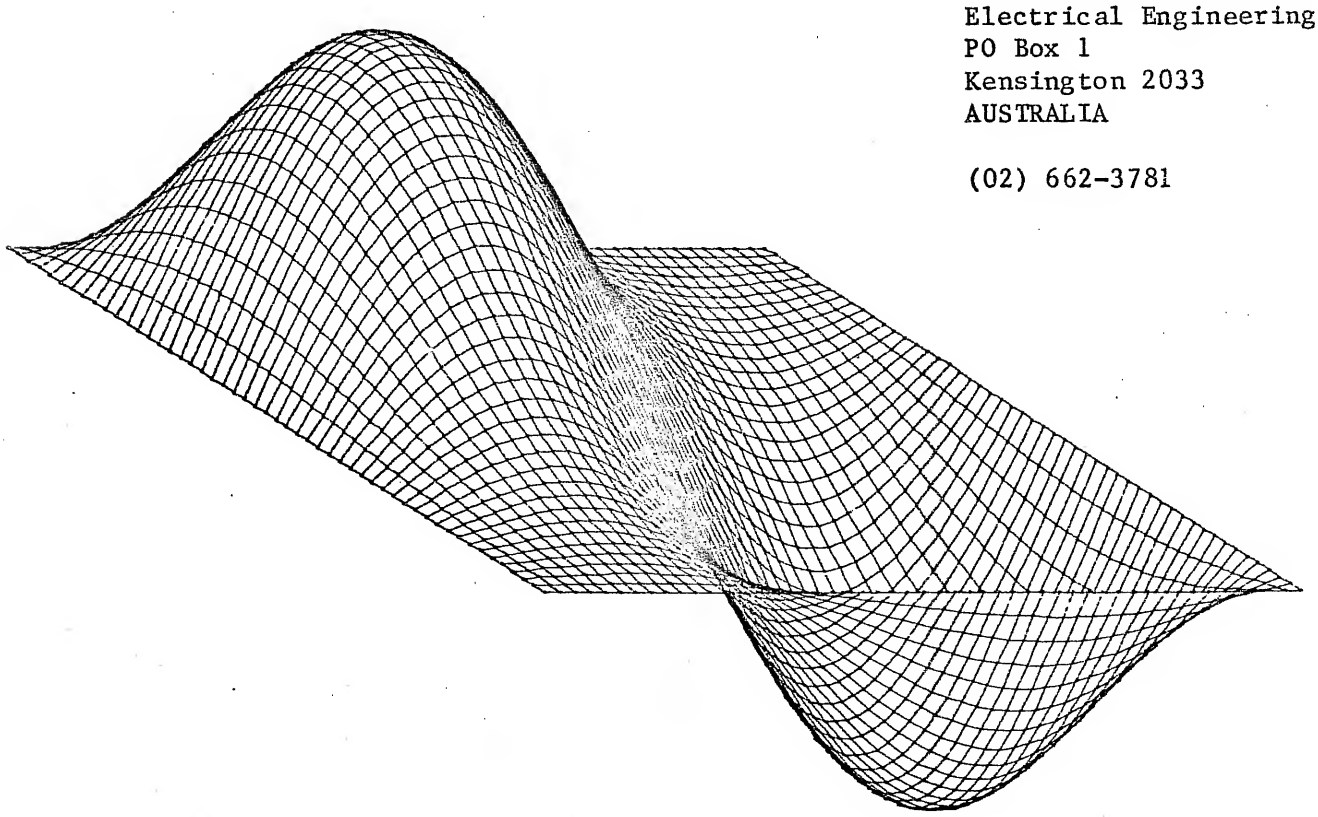
What World Meeting?

As mentioned above, I have nothing to publish about the World meeting. However, Ken McDonell from Monash Uni has requested, and been sent, a set of address labels for all subscribers to AUUGN. Hopefully we will see some detailed information soon.

Pic Of The Month

Peter Ivanov
Dept. of Computer Science
Electrical Engineering
PO Box 1
Kensington 2033
AUSTRALIA

(02) 662-3781



IN REPLY PLEASE QUOTE:

TELEPHONE: 692 3491
692 1122
EXT 3491

UNIVERSITY COMPUTING CENTRE
THE UNIVERSITY OF SYDNEY
NSW 2006

AGENDA - AUUG MEETING 2/7/80

0930	Morning Tea
1000	Introduction
1005	Who, what, where, when
1020	Unix commands for data treament G. Aldis DMS - CSIRO
1040	Typesetting A. Hume AGSM - CSIRO
1100	Simple Programs for Unix/CSIRONET Communication R. Baxter DMS - CSIRO
1120	Share Scheduling Works P. Lauder B.D.C.S. - Sydney Uni.
1140	Simulation of Regression Test Statistics Without Pipes D. Jackett DMS - CSIRO
1200	Lunch at the Terrace Restaurant - Wentworth Bldg.
1400	Discussion - software - future of AUUG - VMS vs UNIX, initiated by Ken McDonell
1430	Unix + C from other places J. O'Brien UNSW
1450	Simulation of Outlier Test Statistics: an Application of Unix Mary Willcox DMS - CSIRO
1510	U.S. Unix Users Meeting R. Elz D.C.S. - Melbourne Uni.
1530	Do it yourself feelthy pictures B. Rowswell UCC - Sydney Uni.
1540	Evans & Sutherland Graphics System Demo + afternoon tea

Site Visits to ChemEng and Basser have been arranged.

Report on Unix Symposium

Vrije Universiteit

Amsterdam

Monday, 24th March, 1980

Alan Mason

ABSTRACT

"Dr. Stephen Johnson, currently of the computer science Research Centre of Bell Laboratories has been a major contributer to the UNIX* appl_cations software, particularly in the area of compilers. He is the author of YACC and the portable C compiler, among other UNIX programs. He is also interested in the portability of C programs in general and Unix in particular. Furthermore, he is building a C machine and adapting a C compiler for the Motorola 68000."

During the week of March 24-28 1980, Steve Johnson visited the Netherlands. The major portion of the week was spent visiting research sites at a number of Dutch Universities but on the first day of his visit he gave a series of 3 Lectures at Vrije University in Amsterdam on the topics of Portable Compilers, Retrofitting Portability to System Software and Language Driven Machine Architecture. These talks and a certain amount of the discussion which followed are described in this report.

The Portable C Compiler

Initially, Steve described some the basic theory of compiler writing, suggesting that not enough people (programmers) paid attentio_ to the known theory (computer science) associated with a problem, and that "the purpose of theory was insight not purely theorems". Taking the basic structure of a compiler to be: Lexical Analysis, Parsing, Optimisation, Code Generation: he showed that only a few of the algorithms of the compiler need be considered for optimality and that most work had been done in the area of Optimal Code Generation (OGC). Refering heavily to recent research of the 70's: Sethi-Ullman, Bruno-Sethi, Aho-Johnson-Ullman, Bruno-Lassagne-Sethi: Steve described the two main approaches to the OGC problem. The use of Directed Acyclic Graphs

(DAGS) and the use of trees to represent the code to be generated. He showed that in the general case DAGS were a bad approach, that tree based solutions were acceptable provided registers were interchangeable, and that in either case the ordering of computation was the most important element. Even using his favoured representation, trees, he showed, with simple examples that tree walks were inadequate and that bottom up dynamic models were of great importance.

In practice, he maintained, OGC was not the real problem since the 'best' solution gives a combination of space time optimality, and that since with any language the code supplied is not completely general, only a limited number of special cases need be optimised. Further, global code optimisation can merely turn linear problems into NP-Complete and "that ordering heuristics can have more payoff than worrying about register allocation".

Compiler theory as yet omits many of the real problems in compiler writing: handling register pairs, assymetric registers, op's with fixed evaluation order, the type and width of operands, conversion operators, bit fields etc.: so that we have a very complicated problem, with no exhaustive theory and many practical issues.

The compiler consists of two passes that together turn C code into assembly code for the target machine, though it may be optionally loaded so that both passes operate as one program. Inter-pass comunication, if required, is made through a non host independent intermediate code (i.e. both passes embody host dependencies). The first pass does lexical analysis, parsing, symbol table maintenance and constructs the parse trees for expressions. Machine depandant portions of this pass generate subroutine prologs and epilogs, code for switches and branches, label definitions etc.. The second pass then takes the generated expression parse trees and using 'Sethi-Ullman' numbers to evaluate the minimum number of scratch registers required to compute a subtree, and a set of templates which descibe target machine instructions, recursively breaks down the tree until it can generate the required code. The machine model used by the compiler assumes the target machine to have a number of registers of two different types, and within each type the registers may be flagged as either scratch or dedicated (stack pointer etc.). Each of the registers in the machine is given a name and number, the number being used as an index into tables that give its type, class, characteristics and usage (the usage of registers is continuously (and globally) monitored lest any of the heuristics used should be incorrect). Further, the compiler writer must specify a set of templates which describe the effect of the target machine instructions on the machine model. Each template is split into a number of sections which specify the subtree to match, desired goal, resources required, instructions to generate and resultant form of the subtree. If an expression tree fails to match directly any template then an attempt is made to rewrite (reshape) it and the process is repeated, a check is built into the compiler to trap excessive recursion.

The portable compiler has an excellent record of generating correct code, the inbuilt cooperation between register allocation, Sethi-Ullman calculation, templates and template rewriting rules build certain

redundancies into the package, and this in turn give a good error detection capability. The compiler can very quickly be running and generating good code for the simple operations and gradually the tables/heuristics built up to fill the gaps. Simply, the problem with compiler writing, in particular of portable compilers, is that they must operate on real machines with a variety of architectures and instruction sets. The saving grace is that "the average expression is extremely simple - get this right, most people will be happy, and many other issues can be handled abysmally without much statistical degradation".

Retrofitting Portability To System Software

In opening Steve described the history of PDP-11 C and its development through B from BCPL. The earliest version was specifically developed for the PDP-11 with types which were in effect the sizes of the available objects. Attempts were made at porting this compiler to IBM and Honeywell kit, but it was a painful experience and led to the generation of inefficient code. A portable C compiler was later developed (Snyder,M.I.T) but, whilst porting this compiler was less painful and the compiler itself was more efficient, compilation was extremely slow. This led to Steve developing his own portable C compiler (henceforth referred to as The Portable C compiler).

The problems to be overcome in porting system software are, to a large extent, overcome by having a compiler which itself is easily ported and by enforcing language constructs which may be easily implemented on any architecture. The most important example of this is in the i/o interface (i/o mechanisms not being specified in the C language definition itself). Further to this preprocessor macro definitions are heavily used to parameterise code, in particular in the area of machine dependent features, and these are stored consistently and uniquely in standard header files.

Within the C language itself this brought about extended data definition capabilities (union, cast, typedef, unsigned) and new data types which were not tied to the word size of the host machine. Improved checking was supported in the form of a separate utility (Lint - picks the bits of fluff out of C programs) which reported 'language holes' (0 used as a null pointer), poor coding (type/argument mismatches between declarations and calls), misuse of variables (incorrect type matching,used before set etc.) and the inevitable trivia such as unused variables. The advantages of splitting compilation and checking into two processes were summarised as being:

Old Programs may still be compiled.
Strong typing model may be ignored if desired.
Compiler can be good and fast at local operations.
Lint can take a global view of programs.
Lint can give warnings intolerable in a compiler.
Lint may be used as a testbed for language changes/restrictions.

Steve maintained that the hard parts of porting the C compiler

revolve around learning the compiler and code generation issues and the design of stack frame, calling sequence and character handling features. Once these issues have been resolved and the compiler set up, it has proved a very dependable package. The compiler is exceedingly suspicious of itself and though it may not generate the 'best' code it will generate 'good' code (i.e. representing the input and acceptably efficient). The time required to port it has varied between 6 months (for a complete novice) to 2 weeks for someone who has done it before! The Portable C Compiler has been adapted to a wide range of machines:

DEC	PDP-11/??,VAX-11/780
IBM	370
HP	300,3000,2100
DATA GEN.	NOVA,ECLIPSE
HONEYWELL	60,6000
SEL	86
ZILOG	Z80,Z8000
INTEL	8085,8086
UNIVAC	V77 (Varian V77)
MOTOROLA	M68000

Once the compiler itself is 'over', porting of UNIX is almost as easy as any other program, except that a limited amount of machine dependent assembler code will have to be rewritten and device handlers generated to suit the new environment:

```
50000 lines of utility software in C
 7000 lines of kernel code in C
      lines of device drivers in C but manuf. dependent
 1250 lines of header declarations
  800 lines of machine dependent assembler
```

When porting the operating system certain problems are encountered; data differences (e.g. byte order), writing/generating a bootstrap file system, original small machine algorithms (memory management, scheduling, i/o etc) may not scale well to large machines; and since operating system debugging is a difficult and tedious task, communications between a UNIX machine and the new host are almost essential.

Language Driven Machine Design

The standard methodology of design (trial design to constraints, performance measurement, improve design, salt to taste and repeat) has failed for computers! This Steve maintains can be attributed to a number of factors:

```
software cost/performance difficult to estimate
programs tend to be rewritten for each new machine
feebdack loop can take years
little mutual understanding between hardware & software
```

In the last 5 years the portability of C and UNIX has become a reality and many C and UNIX applications exist. This in itself justifies looking at architectures which will better suit (i.e. be directly geared to) C. The first step in this process was obviously to look at the type of physical features which a C compiler will use in implementing the normal range of C constructs and form some statistical basis for their necessity. Secondly to guage the least necessary and do away with these. The Portable C Compiler was then used to evaluate the desired changes (i.e. the tables were set up for the new 'trial' design and the compiler run on the same test software) thus reducing the feedback loop from months or years to a number of days. This process was iterated about 12 times and resulted in a number of (redundant) opcodes being dropped and the number of Registers being reduced. Further the design was extended to be a 32 bit machine giving a consequent larger address range, and interestingly, smaller, more efficient code:

standard compiler (cc)	230,000 bytes code
cc with optimiser (cc -O)	215,000
new machine	194,000
new machine with optimiser	?

Possibly, one of the most striking features of the C machine design is that whilst removing a number of general registers an additional, special purpose register has been introduced: the zero register (Z). It was found that the constant '0' is so heavily used by C and UNIX that it was beneficial to be able to access this value quickly and uniquely. This simple addition (very easy to implement!) in conjunction with the completely consistent instruction set can in itself considerably reduce code length.

The eventual design results in having two special purpose registers (zero (z) and stack pointer (sp)), two scratch registers for compiler usage (s1 and s2) and four registers for user assignment (register variables r1 to r4), though statistically a number of these could be removed:

operation	sp	z	s1	s2	r1	r2	r3	r4
R	.004	.079	.351	.042	.074	.052	.025	.002
X(R)	.533	.239	.092	.009	.025	.011	.011	.001
total	.537	.318	.443	.051	.100	.063	.036	.003

An additional scratch register was found to save less than 0.5 per cent of space!

In terms of instruction usage the most common instruction was found (as would be expected) to be the 'mov'. The most common C statement being of the form:

A = B op C

resulting in code of the form:

mov op mov

making, in all, around 44 per cent of the code. The code supplied for the C machine is:

moves	0.41	(moves of arguments-0.18)
branches	0.19	(conditional-0.12, uncond-0.07)
call/return	0.16	
tests	0.10	
+,-,load	0.10	
rest	0.04	

Called upon to show how representative the C test sample was Steve acknoweledged the difficulty in choosing acceptable samples and the even more difficult task involved in justifying the choice but did present a breakdown of the code that had been used:

	Branch	call/save	mov	tst	+/-	etc
C utilities	.19	.16	.41	.10	.10	.04
Pascal	.19	.20	.43	.11	.04	.03
TPC	.18	.21	.41	.09	.05	.06
--SYS	.17	.17	.40	.09	.08	.11
sh	.19	.20	.41	.09	.06	.06

and stated that over half of 'etc' consisted of addressing operations. Finding it difficult (as everyone has) to justify certain initial design decisions Steve resorted (in best British fashion) to an anecdote: Once upon a time, in a little known region of the Americas, a computer manufacturer, mindfull of the efficiency of his software (not to mention his small machine) decided that a statistics of instruction execution would be useful. Caryying this out with the true zeal of a commercial concern a bus monitor was installed and data logging equipment used to accumulate this very necessary information on magnetic tape for later processing. Many reels of tape later the system was brought to a halt and the tapes processed: nothing unusual in the results! A flash of inspiration (or was it a head crash?): process the tapes and find out if any instructions occur frequently in pairs. Minions were dispatched, programs written and tapes spun once more. Did they? - yes! Sucess, a new operation code was created by merging these ops. The microcode of the machine was changed, the compilers were modified to use the new marvel code and all the software was recompiled. Ready to roll again the bus monitor was installed and all the data logged. No Change! What was wrong said the enterprising programmers? Why do our programs not run faster? Do our programs use this instruction? What does use this instruction? Answer - the idle loop!!!

After the Bell

The visit had been arranged and coordinated by Andy Tanenbaum and knowing that although Steve would be in Holland for the rest of the week a number of the attendees had travelled far (Scotland,France,Germany) and would have to head homewards, he very thoughtfully arranged that the non-nationals could get together with Steve for an evening meal. From this session, as well as the well known discussions condemning certain

computer manufacturers, came some very useful information and ideas, some of which I give here. The remainder, that which Steve would not like to be quoted on, will probably appear in the (insubstantiated) gossip column!

Much of the discussion centered around the new Motorola M68000 which Steve seemed very pleased with. The C compiler had already been ported to it and UNIX was under development. Bell has already ordered 200 of these items and intends configuring the as a multiprocessor and developing a multiprocessor UNIX for them. Further, since this very powerful machine (about the same as an PDP 11/45 but with 32 bit word capability) is so very cheap and will shortly be available packaged as a system with Winchester technology disks, Bell were about to announce a 'personal computing' licence for UNIX at about the same price as an academic/research licence ($300 ?).

In an attempt to allow grater portability of ultilities, Bell Research and their legal advisors are currently investigating the possibility of releasing the Portable C compiler as a separate item (i.e. C in the public domain). This step, it is hoped would make C much more readily available, allow greater development of general utility programs, and of course give C a much better chance of competing in the ever expanding language market.

Asked about Bell's attitude to releasing staff to speak at 'foreign' meetings Steve said that, like any research establishment, they wanted to feel that they were getting some return from the trip. Future talks would only be countenanced if there were a reasonable assurance that the participant would also be able to visit local research centres and hopefully gain ideas and information.

Alan Mason

UKUUG

UNITED KINGDOM UNIX USER GROUP

R.A.Mason
Department of Electrical and
Electronic Engineering
Mountbatten Building
Heriot-Watt University
31-35 Grassmarket
Edinburgh

UNIX* V7 strip-down

Unix version 7 as distributed by Bell Labs is not intended for use on small (non separate i/d) PDP-11's. Jim McKie (Heriot-Watt University) in conjunction with Dave Rosenthal (Edinburgh University) and Colin Prosser (SRC Rutherford) have developed a stripped down version which should run on any PDP-11 and has in fact been tested on 11/34, 11/40, 11/45, 11/60 though the machine support (backup) has been tested on everything (except an 11/23) with and without hardware floating point. This package will be available on 2400 ft 800 bpi magnetic tape from the 1st of July.

The package differs from that distributed by Bell in a number of ways:

1) It can be booted from tape on to RK05 and RL01 systems

2) New teletype handling subsystem (tty.c,dz.c,dl.c) to UK standard.

3) Additional handlers include single drive SI, paper-tape, Boston style RK05, glasgow semaphores.

4) Ifdef'd mods allow the system buffers to reside in user space and thus allow the system itself to be much larger. The default system (without this mod) can support about 3 device drivers (e.g. DZ,RL,TM) with about 14 buffers and very little space to spare.

*UNIX is a Trademark of Bell Laboratories.

5) Certain of the utilities originally required seperate i/d in order to run. These have been reduced in size (either by table squeezing or code stripping) to get executable versions. Falling in to this category are:

adb	awk	dcheck	dd	fgrep	icheck
lex	ncheck	refer	restor	sort	spell
tar	tbl	yacc			

6) Other utilities had to be positively upgraded to reach the standard to which we have become accustomed or simply to get them to work:

em - superset of 'ed', features as per qmc 'em'
cu - call up another machine (did not work)
plus known US and Australian bug fixes

7) As yet certain utilities cannot be usefully run on small machines:

lint f77 struct

If you want to get your hands on this system then write to me at the above address sending a copy of your Unix V7 licence and a cheque for 15 pounds to cover cost of tapes, post and packing. Please make cheques payable to "UUG - Heriot-Watt University".

'Make' and Libraries

Dave Rosenthal

EdCAAD Studies
Dept. of Architecture
Edinburgh University

9th July, 1980

Version 7's `make' utility is almost worth the effort of bringing up the new system, all on its own. Writing `makefiles' for all normal ways of building programs from source files is very easy; the built-in rules do most of the work for you, and the directories `/usr/src/cmd/*' have many useful examples.

However, it is not immediately obvious how to write a `makefile' to update a library without re-compiling source files which have not changed. The example below shows the way we do it; other ways may exist. Assuming that the library is called `lib.a', that it contains the object code from `s0.c', `s1.c', ..., `sN.c', and that `s0.c' uses `s0.h' and so on, the makefile looks like this:-

```
lib.a : s0.c s1.c ..... sN.c
        -TMP=/tmp/make$$$$ ; \
        trap 'rm -f $$TMP $$B.o; exit' 1 2 3 15 ; \
        if test -f $@ ; \
        then \
                cp $@ $$TMP ; \
        fi ; \
        for A in $? ; \
        do \
                echo $$A: ; \
                B=`basename $$A .c` ; \
                cc $(CFLAGS) -c $$B.c ; \
                ar r $$TMP $$B.o ; \
                rm $$B.o ; \
        done ; \
        mv $$TMP $@

s0.c :  s0.h
        touch $@

        <and so on>
```

The following points are worth noting:-

1. The commands to update the library do not refer to any specific file names, so they can simply be copied into the 'makefile' for a new library.

2. The trailing semi-colons and backslashes are required; 'make' gives each command line to a separate shell, and so must be fooled into thinking that this is a single line.

3. The profusion of dollar signs results from the need to quote the dollars intended for the shell through 'make'.

4. The initial '-' on the update command ensures that the success or failure of the 'test' command is seen by the 'if', and not by 'make'.

5. The use of a temporary copy of the library being updated ensures that interrupting a 'make' leaves things in a sensible state (i.e. as they were before the 'make' started). We haven't yet found a simple way of interrupting a library 'make' so that a subsequent 'make' starts where the interrupted one left off.

If anybody can provide a neater way of doing this, please let us know. Another useful addition to this technique would be a 'sed' script to generate a 'makefile' for each of the system libraries by reading the code, and generating a dependency line for each

```
#include        "......"
```

line. It should also generate the dependency lines like

```
LIB2 : rk.c
```

by reading 'mklib'. I have a script that partly works, but the amount of hand fiddling its output needs to be really correct is excessive.

RESET/SETEXIT REVISITED

Bruce Anderson, University of Essex

There is a definite need for nonlocal jumps/exits in writing C programs, but the RESET/SETEXIT facility provided is pretty awful - it's got bugs in it, and anyway allows only one state to be the current target of a RESET. A much better style (due to Peter Landin) can be very simply implemented by:

```
        .globl _catch, _throw, cret

_catch:
        mov     r5, r0
        rts     pc

_throw:
        mov     4(sp), r0
        mov     2(sp), r5
        jmp     cret
```

The two routines CATCH and THROW now work as follows. CATCH is a function of no arguments, returning an object which we will call a ball (!). If this ball is THROWn with a value, then the function application in which the ball was created (by CATCH) is exited with that value. As an example, in

```
foo(anarg)
   {    int theball; theball = catch();
        ...; baz(theball); ...;
   }

baz(aball)
   {    ...; throw(aball, 77); ...;
   }
```

then if the call to THROW in BAZ takes place, the corresponding call of FOO will exit with value 77. A typical use is to exit from some very deep recursion when you've finally found what you are looking for, or to get back to some previous state when some signal is received. Of course it's only the control state that is restored, and not all the variables that may have been affected by assignments after the ball was created!

Anyway, I think this is a cleaner facility than Harold Thimbleby's LEAVE (see Software - Practice and Experience not long ago), as

* it's structured - someone has to give you a ball before you can throw it back
* balls are proper objects
* no searching of the stack is done to find where to exit
* it works with I&D separation

A simple modification would check that the ball was round i.e. that it corresponded to the exit of some currently active procedure. Of course, it might be a forgery, but that leads us into more remote territory.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE
UNIVERSITY OF EDINBURGH

FORREST HILL
EDINBURGH EH1 2QL
Telephone 031-667 1011 Ext:

Head of Department
J.A.M. HOWE

Date: 31 March 1980

Ref:

Thank you for your enquiry about Unix POP-2 and Prolog systems for use on a DEC PDP-11 under Unix. The systems were written in the Department of Artificial Intelligence of the University of Edinburgh, which is also the sole distributor. The systems correspond closely with the standard language specification, and a number of minor improvements have been made.

Both systems are written in assembly language, and run under the UNIX Version 6 operating system. POP-2 occupies 8K words (shared) and 2K words (unshared), and Prolog occupies 4K words (shared) and 5K words (unshared). In both systems, the remainder of the 32K word user address space is used for workspace and storage of user data structures, and is obtained from and released to Unix as required. Both systems feature compacting garbage collectors.

The POP-2 and Prolog systems are distributed together, either on 9-track 800 bpi tape written in Unix tp format, or on RX01 floppy discs. The RX01 discs are written as UNIX or RT-11 filesystems, and the source of a standard RX01 device driver is included. All source files, documentation, and installation instructions are provided free of charge, but a small charge (thirty pounds within the United Kingdom, fifty pounds abroad) will be made for carriage and media. Do not send us tapes or floppies. Any media recieved will be treated as an unsolicited gift.

Before supplying copies of the systems, we require that potential users complete an agreement restricting their use to non-commercial academic and educational purposes. We also require that the potential user has a UNIX licence.

If you would like a copy of the UNIX POP-2 and PROLOG systems, write to us at the address given below ensuring that you enclose the full name and address of the institution that wishes to enter into the agreement with us. We will then send you a Request Form and the Software Agreement for you to complete. In the Request Form you will be asked to specify the desired medium of distribution.

We look forward to your reply.

The Software Secretary
Department of Artificial Intelligence
University of Edinburgh
Forrest Hill
EDINBURGH EH1 2QL
Scotland

ETH EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE
ZÜRICH

Institut für Informatik

Clausiusstrasse 55
Telefon 01 326211

Postadresse:
Institut für Informatik
ETH-Zentrum
CH-8092 Zürich

April 1980

MODULA-2

You have previously expressed your interest in our work on the programming language Modula-2. I am now happy to enclose a defining report and to announce that we will be able to make available a Modula-2 implementation for the PDP-11 computer in June (exactly 10 years after our first release of Pascal).

This system includes a compiler, a linker, a small resident "operating system" with a loader, a set of utility modules, and - if everything goes well - a debugger. We operate under RT-11 on a 28K computer with an RK-05 disk. The extended instruction set (EIS: multiply, divide) is required. FIS is needed, if the type REAL is used.

The release will include all software in compiled form, and additionally all utility modules (in particular those which refer to the operating system) in source form. This should allow adaptation to other operating systems with minimal effort. An adaptation to UNIX is already in preparation. The system also operates on the LSI-11.

If you are interested in receiving the system, please return this form. The package will include the software on an 800 bpi tape and documentation. The handling charge is SFr. 350.-- and includes the tape and the documentation.

☐ I herewith order the Modula-2 system for RT-11 and enclose a check of SFr. 350.-- .

☐ I would like to be notified when the Modula-2 system for UNIX is available.

Name and address:

Sincerely,

N. Wirth

Prof. Niklaus Wirth

A PORTABLE COMPILER FOR THE PROPOSED ISO STANDARD PASCAL LANGUAGE

Andrew S. Tanenbaum
Johan W. Stevenson
Hans van Staveren

Computer Science Group
Vrije Unversiteit
Amsterdam, The Netherlands

We are pleased to announce that the May 1980 release of our portable Pascal compiler conforms to the recently proposed ISO standard. The source language accepted is the full language, including procedure and function parameters, conformant arrays, local files, etc. The compiler produces an intermediate code called EM-1, which can then be assembled to binary EM-1 code and interpreted, or translated to the assembly language of the target computer by a back end program, and then executed directly by the target computer.

There are three distribution tapes available:

(1) for the PDP-11 under UNIX* version 6,
(2) for the PDP-11 under UNIX version 7, or
(3) for non PDP-11 or non UNIX machines.

The UNIX versions supersede the (non ISO) compiler on the 4th UNIX distribution tape. The UNIX versions contain a program pc (analogous to cc). The command **pc prog.p** compiles the file prog.p to EM-1 binary code for subsequent interpretation, whereas the command **pc -C prog.p** compiles the file prog.p to a true executable PDP-11 binary program (a.out file). In addition, a library containing all the UNIX system calls is provided, so seek, pipe, fork, exec, etc., can all be called from Pascal programs. It is also possible to call user supplied C functions from Pascal programs and vice versa. The system contains extensive facilities for debugging, assertion checking, tracing, profiling, separate compilation, double precision arithmetic, etc. It is intended as a serious production system and not a toy. Complete documentation is included. The system has been designed in a modular way, to allow it to be modified to become a cross compiler for other machines. In fact we are actively working on this now, and will report on it shortly.

The tape for non UNIX installations is identical to the UNIX tape, except that a few proprietary Bell Laboratories programs have been removed. The compiler proper is just a Pascal program, and hence is highly portable. Some of the other programs, e.g., the EM-1 assembler-loader and the EM-1 to EM-1 peephole optimizer are written in standard C, and hence are also reasonable portable. The libraries are largely written in EM-1 assembly language, and are thus also portable.

Although the software itself is free, there is a $50 charge for the tape and postage. Inquiries from Europe should be directed to us. For the convenience of people in North America, Intermetrics, Inc., 701 Concord Ave., Cambridge, Mass. 02138 has agreed to distribute the tapes there. Intermetrics has moved an earlier version (non ISO) of the system to PWB.

*UNIX is a Trademark of Bell Laboratories.

DEWAR INFORMATION SYSTEMS CORPORATION 221 WEST LAKE STREET OAK PARK, ILLINOIS 60302 USA 312.524.1644

June 6, 1980

Ian Hayes
School of Elec Eng & Computer Science
THE UNIVERSITY OF NEW SOUTH WALES
P. O. Box 1
Kensington
New South Wales
AUSTRALIA 2033

Dear Mr. Hayes,

The UNIX version of VAX SPITBOL has been completed, however, we do not as of today (June 6) have distribution tapes in stock. We have been informed that tapes should be arriving before the end of June.

The VAX SPITBOL license is a separate license from that of the PDP-11 SPITBOL license. The cost for an educational institution is $795.00 plus a media charge of $10 for the 800 BPI MAGtape.

If I can be of any additional help, please let me know.

Yours sincerely,
DEWAR INFORMATION SYSTEMS CORPORATION

Susan Anderson Deets

Susan Anderson Deets
Business Manager

SAD/je

SPITBOL

DEWAR INFORMATION SYSTEMS CORPORATION

DATA SHEET
OCTOBER 1979

VAX SPITBOL NOW AVAILABLE

SPITBOL for the VAX 11/780 and PDP-11 series of minicomputers is a full implementation of the SNOBOL4 programming language. VAX SPITBOL and PDP-11 SPITBOL are completely compatible with other implementations of MACROSPITBOL (including those on the CDC 6600/CYBER, DEC 10, MODCOMP and INTERDATA computers.) It will also accept most programs written for the SNOBOL4 and SPITBOL systems on the IBM370 without modification. The VAX SPITBOL translator runs in the native instruction mode of the VAX computer and takes full advantage of the extended addressing and virtual memory capabilities of the machine.

Most SNOBOL4 programs will operate correctly when compiled and executed under SPITBOL. There are a few minor incompatibilities and some features of SNOBOL4 have not been implemented (notably the QUICKSCAN mode of pattern matching and the capability to redefine standard system functions and predefined operators). On the other hand, SPITBOL contains many useful enhancements to the SNOBOL4 language, including an expanded TRACE function for improved debugging of application programs, and several additional built-in functions for sorting data items and facilitating the output formatting and comparison and manipulation of strings.

UPDATE POLICY

Updates to future, enhanced versions of SPITBOL will be available to SPITBOL licensees at a nominal cost (currently $75.)

DISTRIBUTION MEDIA

The distribution medium is normally 9 track, 800 BPI MAGtape (at $10). Other media may also be available on special request.

LICENSE

VAX SPITBOL and PDP-11 SPITBOL license fees below are for a primary license. Secondary licenses (ordered through the same purchasing organization) are available for 50% of the primary license fee.

ORDERING INFORMATION

VAX 11/780* for a single CPU

VAX SPITBOL (perpetual license) includes source, object, and manual

Commercial institutions	$1195.00
Educational & non-profit	795.00

Operating System-specify on purchase order

VMS*

UNIX(tm)

VAX SPITBOL Manual 7.50

PDP-11* for a single CPU

PDP-11 SPITBOL (perpetual license) includes source, object, and manual

Commercial institutions	$795.00
Educational & non-profit	595.00

Operating System-specify on purchase order

UNIX(tm)

RSX-11m*

RSTS*

DURESS+

PDP-11 SPITBOL Manual 7.50

Minimum PDP-11 configuration is an 11/34 with EIS. SPITBOL itself occupies about 16k words. A full 32k partition should be available.

+DURESS is a tradename of DATALOGICS, Inc.

(tm)UNIX is a tradename of BELL TELEPHONE LABORATORIES

* are tradenames of DIGITAL EQUIPMENT CORPORATION

Phone 312.524.1644

DEWAR INFORMATION SYSTEMS CORPORATION
221 West Lake Street Oak Park, Illinois 60302 USA

UNIX FROM THE POINT OF VIEW OF A COMMERCIAL USER

INTRODUCTION

Until recently L.E.R.S. was the only commercial unix licence holder in Europe. Now I believe that we have been joined by Inmos (who I believe are present at this meeting) if not by others. I know of at least one other major UK company who has considered using Unix the project finally being abandoned for non technical reasons (i.e. not because they found something better). We have been approached by four or five major French companies to advise on unix. I would be very happy to hear of any other commercial users in Europe. Remember of course that unix started in a relatively commercial environment.

L.E.R.S. has been a commercial user of unix since the end of 1976 and perhaps because of our relatively long experience with unix and undoubtedly because of our scarsity value I as head of the computer group at L.E.R.S. have been asked to say a little to you about our experiences with Unix in a relatively commercial environment.

L.E.R.S.

First what is L.E.R.S.? L.E.R.S. is Laboratoires d'Etudes et de Recherches Synthelabo. What is that? Synthelabo is the fourth largest French drug and hospital products company. L.E.R.S. is the drug research end of Synthelabo. L.E.R.S. employs at present about 400 people of whom about 30% are non French; these people range from chemists through biologists to medics and pharmacists and even to computer scientists. Our laboratories are situated on two sites one in southern Paris the other a few miles away in the southern suburbs of Paris. We also have a few people at London, at Tours and elsewhere.

HISTORY OF THE COMPUTER GROUP AT L.E.R.S.

Back in 1976 when there were basically no computer facilities at L.E.R.S. one research group decided it wanted a computer to automate its cardiovascular laboratories. The head of that group had worked in the USA and had there come across PDP11s and Unix fanatics. When he finally persuaded L.E.R.S. to buy a computer he managed to get something rather bigger than his department needed so that eventually it might be able to provide some sort of computing service to the rest of L.E.R.S. The result was that a PDP 11/70 and a unix version 6 licence were purchased. Unix was installed for us by three young Americans acting as consultants; one of these three still acts as consultant for us and is responsible for most of the major modifications that have been made to our version of Unix. At the end of the installation period one programmer was taken on. Overall responsibility for the system rested initially with the cardiovascular group, then with the statistics and pharmacokinetics group and finally the groupe informatique was set up. I arrived at L.E.R.S. as chef du groupe informatique in January 1978 with no Unix experience, no PDP 11 experience but with lots of biomedical computing experience.

THE HARDWARE AT L.E.R.S.

I think we would be classed as a fairly big Unix site. We have a PDP 11/70 with 96K DEC memory and 128K Intel memory. Two discs one an 80 Mbyte RP04 from DEC the other a 300Mbyte CDC disc supplied by System Industries. A DEC mag tape TU10. 3 Printronix printer/plotters one connected by a parallel interface the others connected by serial interfaces. A Benson graph plotter working through an RS232C interface. A Qume daisy-wheel printer. An LA36 DECwriter as console device. We have a 16 channel ADC system connected directly to the PDP 11/70 (AR11); our version of Unix was substantially modified in the early days to support one real-time user recently however this type of work has been done using microprocessors connected to RS232C type interfaces. We have currently 35 serial RS232C lines (DZ11s and DL11s) to which are attached of the order of twenty terminals (mainly Perkin Elmer Foxes), four LSI11 microprocessors, two printer/plotters, the daisy-wheel printer and the Benson plotter. Recently we have purchased a KMC11A auxilary processor to try to reduce he interrupt loading caused by all the character mode interfaces. We hope that the KMC11A will enable us to do eventually attach more terminals than would otherwise be possible, I must admit that the software for it is not yet fully operational; if anyone wants it though we have an assembler for the KMC11A microcode.

We currently have five LSI11/1 microprocessor systems attached to the computer and we will have about four more LSI11/1 or 11/23 systems by the end of 1980. These systems are used for real-time data acquisition in laboratories. The microprocessor systems are assembled by our electronics group who incorporate amplifiers, ADCs, etc.

Our main communications link between our two sites consists of a 72 kilobaud link working over a 50 KHz bandwidth line. We have also done some communication at 300 baud over the public switched network and will have to again. For communications with less important outstations we are putting in 9600 baud or 19200 baud base band links. The 50 kilobaud link is multiplexed giving at present 8 channels and in the near future 12 channels. The slower links will be single channel initially but eventually will have multiplexors added.

OUR VERSION OF UNIX

Our version of Unix is a heavily modified version 6 which in some respects is approaching a version 7. We are awaiting our version 7 licence. The biggest modification to our version of unix was the introduction of a real-time partition. This means that one user can use the ADC (AR11) which is directly connected to the PDP 11/70 whilst others users carry on in the normal way (but with a very much slower response dependant on sampling rate). This modification was necessary for us but I would in no way recommend it to other users. Now we are trying to do all real-time work on the LSI11s. At the moment we are experimenting with an LSI11 on parallel DMA interfaces which when it works should finally remove the need for the real-time segment and allow us to go to a purish version 7. Other major system software developments have been in the realm of communications software for the LSI11s (talk to me afterwards if you want more details) and a mini O.S. for the LSI11/1s with floppy disks.

APPLICATIONS AT L.E.R.S.

They are many and varied. Perhaps they can be split into four categories: firstly laboratory automation and signal processing, secondly database projects, thirdly word processing and fourthly other projects. The PDP 11/70 was installed with a view to automating certain cardiovascular laboratories. To date we have automated three cardiovascular research laboratories and are working on various EEG, bahaviour and blood flow laboratories. All of our database projects are centred around a highly flexible and relatively powerful database generating system developed for us initially by our American consultant and further by my colleague Alastair Adamson who is here with me today. If anyone would like any further details of this database generating system thatwe have I suggest you talk to Alastair over lunch. The databases that we have actually implemented using this system cover a wide range of subjects; information systems to assist project control, name and address lists, scientific references, laboratory results, payments due to service companies, preparation of bibliographies, details of electronic equipment used by L.E.R.S. a library index and chemical databases. The word processing that we do is mainly the preparation of internal reports using ed, em, ex, nroff, neqn, etc. This word processing use is expanding all the time. A large number of statistical and pharmacokinetic programs are run on our system, we have developped sophisticated graphics programs for use with our Benson plotter, the Printronix printer/plotters and various Tektronix terminals, we do EEG analysis from tape recordings made during clinical drug trials, we are doing a little PERT and starting to use programs to assist chemists. In other words our applications are wide ranging and I can assure you there are at least as many new ideas in the pipeline.

STAFF

What staff do we have to do this work. Answer myself and five other people. One person, the original programmer is rsponsible for system problems and certain aspects of laboratory automation. He has an assistant who is implementing further real-time systems in the cardiovascular department. Someone else is responsible for our activities at our site near Paris and is particularly involved with laboratory automation. We have one person Alastair Adamson (who was brought up on Unix at Sterling) who is responsible for all the database developments. And we are just taking on a Dutchman to replace an Italian who was developping programs used to analyse EEGs from clinical trials.

WHAT ARE OUR PLANS FOR THE FUTURE?

We hope to stick with unix for a reasonable length of time. Our PDP 11/70 will be overloaded in the not too distant future which probably means a VAX or another PDP 11/70 within the next two years. There will undoubtedly be more microsystems some of the 11/23s hopefully runnning a version of unix. We think we will always have a central time-sharing facility but that more and more power will be distributed where it is needed. We may buy non unix like systems for special purposes, e.g. databases of chemivcal structures, or word processing (in this case just to prove that unix is best). Our policy at the moment is to stick with genuine DEC processors not necessarily bought from DEC and otherwise to minimise our number of different types of kit. We have a big enough mixture of kit already and do not intend to buy other types unless there is a really very strong reason. Occasionally it is suggested that we should be moving towards IBM kit; I might accept that when they offer unix.

CONCLUSIONS

We are happy with unix; we have been happy with unix now for three and a half years.

AU REVOIR

I hope this talk about the French connection has interested you. You may be interested to know that there is now at least one French academic licence holder who until he gets a machine to go with his licence is (with the consent of Western Electric) using our computer.

Ian R. Perry 26/3/80

The following sheet headed 'Square Brackets Competition - Results' comes from the UK newsletter and demonstrates something, although I am not quite sure what. All to often have people asked questions similar to that at the top of the page and received answers similar to those appearing after.

The fact that the obvious grep command:

```
grep '[\[\]]' filename
```

fails demonstrates an bug in grep which should be corrected.

But people can code round any obstacle as demonstrated by the solutions given. Why is it that when we are confronted with a basic programming error we do not think 'this is a bug and I will fix it' but go off and beat some other utility into submission.

The third correct solution demonstrates with ultimate finality how people can become so familiar with these black and blue programs that exploiting one idiosyncrasy to get around another is taken as fair play.

Good grief !!!!

Locally grep works as it should.

Peter Ivanov
UNSW

SQUARE BRACKETS COMPETITION - RESULTS

"All I want is a shell file which implements a command to print out all the lines of a file which contain a left or right square bracket (or both, of course)." [newsletter 6, p16]

The syntax for alternatives in GREP/ED uses square brackets, so some care will obviously be needed, i.e. GREP "[[]]" fails. The obvious solution is GREP "[\[\]]" but this fails too. If only those Unix utilities hadn't been written so QUICKLY! I received three correct solutions:

1 (and the winner) from Ian Cottam in York

```
(grep -n "\]" $1; grep -n "[" $1) ^ sort -n ^ uniq > tmp
ed - tmp
g/^/s/[^:]*://p
q
rm tmp
```

This is a brilliant answer in perfect Unix style - uses several tools, some pipes, and flags you never knew existed.

2 from Tony Walsby at QMC

```
ed $1 > /dev/null
1,$s/^/#/
g/\[/s/^/#/
v/^#/s/.*\]/#&/
v/^#/d
g/^#/s///
1,$w tony1
q
cat tony1
rm tony1
```

Quite different - the "hammer one tool to death" and " private language" approaches welded together into an irresistible monolith.

3 from Colin Taylor at Westfield College, and several others

```
grep "[][]" $1
```

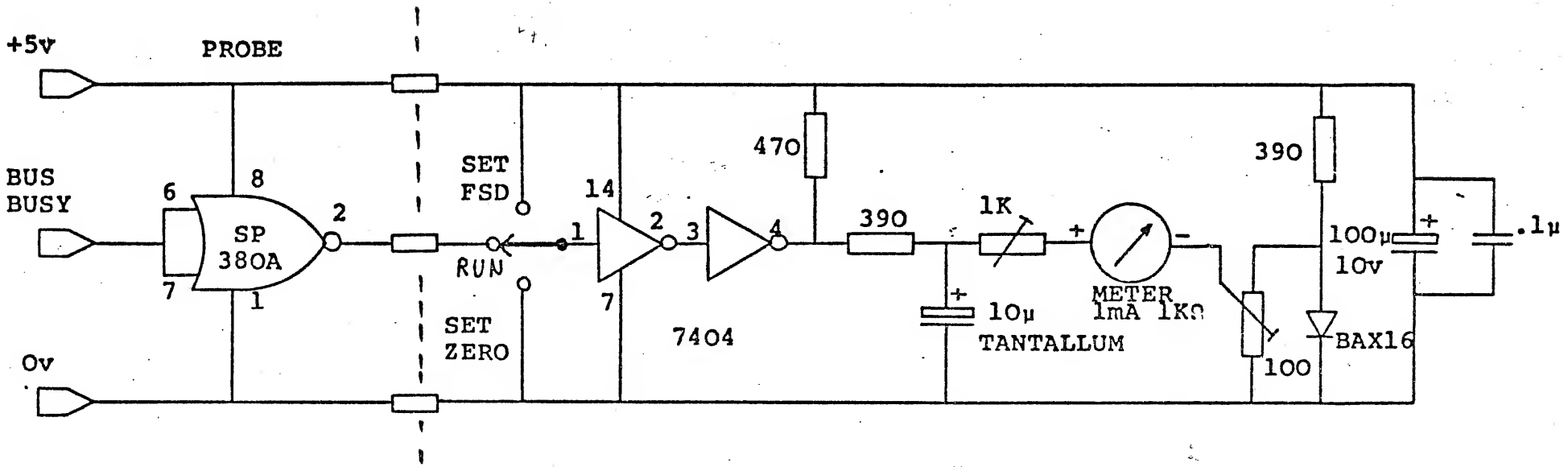
This is the Unix-wizard/kernel-hacker's master stroke - "Zen and the art of string-manipulation".

> "... It may not be general knowledge, but in the editor syntax [abc...] for character classes, the first character, a, is not checked at all, and subsequent characters bc... are only tested against "]", so the grep one-liner ..."

was the explanation as nicely put by Andy Walker from Nottingham.

Bruce Anderson

THRUST METER



The SP380A is mounted in a "probe" close to the bus to reduce capacitative loading, and is then connected by a screened cable to the meter assembly.

To calibrate:

set switch to "set zero", set 1K to minimum resistance, adjust 100 for zero on meter .
set switch to "set fsd", adjust 1K for fsd on meter

[Yes, the meter reads the fraction of the time that the bus is busy, and it's very useful. At Essex an opto-coupled extender drives remote meters in several labs! The main meter on the 11/45 is a beautiful old 9" diameter model reading THRUST LBS x 1000 from 0 to 10.]

Designed originally by Pete Madams (Essex), modified by J Feenstra (Nijmegen), and this drawing executed by Paul Griffith (Essex).

UNIX NEWSLETTERS

are edited by the following bunch of loonatics:

Peter Ivanov
Computer Science
Electrical Engineering
University of New South Wales
PO box 1
Kensington 2033
Australia

Bruce Anderson
UK UNIX SIG Newsletter Editor
Electrical Engineering Science
University of Essex
COLCHESTER C04 3SQ
England

Newsletter Editor
Usenix Association
Box 8,
The Rockefeller University
4230 York Ave.
New York, NY 10024
USA

Neil Groundwater
Software Tools Newsletter Editor
Analytic Disciplines Inc.
8320 Old Courthouse Road
Vienna, VA 22180
USA

Newsletter Editor
Login West
PO Box 584
Menlo Park
California 94025
USA

Mike Tilson
Canadian UNIX Sig Newsletter Editor
Human Computing Resources Corp.
40 St. Mary Street
Toronto, Ontario
M4Y 1P9